

THE LOCAL LEMMA IS TIGHT FOR SAT

H. GEBAUER, T. SZABÓ, AND G. TARDOS

ABSTRACT. For every $\epsilon > 0$ and large enough k we construct unsatisfiable k -CNF formulas where every clause has k distinct literals and every variable appears in at most $(\frac{2}{e} + \epsilon) \frac{2^k}{k}$ clauses. Using a result of Berman, Karpinski and Scott we also show that our result is asymptotically best possible: every k -CNF formula where every variable appears in at most $\frac{2^{k+1}}{ek} - 1$ clauses is satisfiable. The determination of this extremal function is particularly important as it represents the value where the k -SAT problem exhibits its complexity hardness jump: from having every instance being a YES-instance it becomes NP-hard just by allowing each variable to occur in one more clause.

The asymptotics of other related extremal functions are also determined. Let $l(k)$ denote the maximum number, such that every k -CNF formula with each clause containing k distinct literals and each clause having a common variable with at most $l(k)$ other clauses, is satisfiable. We establish that the bound on $l(k)$ obtained from the local lemma is asymptotically optimal, i.e., $l(k) = (e^{-1} + o(1)) 2^k$.

The constructed formulas are all in the class MU(1) of minimal unsatisfiable formulas having one more clause than variables and thus they resolve these asymptotic questions within that class as well.

Institute of Theoretical Computer Science, ETH Zurich, CH-8092, Switzerland; mail: gebauerh@inf.ethz.ch.

Department of Mathematics and Computer Science, Freie Universität Berlin, 14195 Berlin, Germany; mail: szabo@zedat.fu-berlin.de.

Simon Fraser University, Burnaby BC, Canada and Rényi Institute, Budapest, Hungary; mail: tardos@cs.sfu.ca.

1. INTRODUCTION

The satisfiability of Boolean formulas is the archetypical NP-hard problem. Somewhat unusually we define a k -CNF formula as the conjunction of clauses that are the disjunction of exactly k distinct literals. (Note that most texts allow shorter clauses in a k -CNF formula, but fixing the exact length will be important for us later on.) The problem of deciding whether a k -CNF formula is satisfiable is denoted by k -SAT, it is solvable in polynomial time for $k = 2$, and is NP-complete for every $k \geq 3$ as shown by Cook [3].

Papadimitriou and Yannakakis [18] have shown that k -SAT is even MAX-SNP-complete for every $k \geq 2$.

The first level of difficulty in satisfying a CNF formula arises when two clauses share variables. For a finer view into the transition to NP-hardness, a grading of the class of k -CNF formulas can be introduced, that limits how much clauses interact locally. A k -CNF formula is called a (k, s) -CNF formula if every variable appears in at most s clauses. The problem of satisfiability of (k, s) -CNF formulas is denoted by (k, s) -SAT.

Tovey [24] proved that while every $(3, 3)$ -CNF formula is satisfiable (due to Hall's theorem), the problem of deciding whether a $(3, 4)$ -CNF formula is satisfiable is already NP-hard. Dubois [5] showed that $(4, 6)$ -SAT and $(5, 11)$ -SAT are also NP-complete.

Kratochvíl, Savický, and Tuza [15] defined the value $f(k)$ to be the largest integer s such that every (k, s) -CNF is satisfiable. They also generalized Tovey's result by showing that for every $k \geq 3$ $(k, f(k) + 1)$ -SAT is already NP-complete. In other words, for every $k \geq 3$ the (k, s) -SAT problem goes through a kind of "complexity phase transition" at the value $s = f(k)$. On the one hand the $(k, f(k))$ -SAT problem is trivial by definition in the sense that every instance of the problem is a "YES"-instance. On the other hand the $(k, f(k) + 1)$ -SAT problem is already NP-hard, so the problem becomes hard from being trivial just by allowing one more occurrence of each variable. For large values of k this might seem astonishing, as the value of the transition is exponential in k : one might think that the change of just one in the parameter should have hardly any effect.

The complexity hardness jump is even greater: the problem of (k, s) -SAT is also MAX-SNP-complete for every $s > f(k)$ as was shown by Berman, Karpinski, and Scott [2] (generalizing a result of Feige [7] who showed that $(3, 5)$ -SAT is hard to approximate within a certain constant factor).

The determination of where this complexity hardness jump occurs is the topic of the current paper.

For a lower bound the best tool available is the Lovász Local Lemma. It does not deal directly with number of occurrences of variables, but rather with pairs of clauses that share at least one variable. We call such a pair an *intersecting pair* of clauses. A straight forward consequence of the lemma states that if every clause of a k -CNF formula intersects at most $2^k/e - 1$ other clauses, then the formula is satisfiable. It is natural to ask how tight this bound is and for that Gebauer et al. [9] define $l(k)$ to be the largest integer number satisfying that whenever all clauses of a k -CNF formula intersect at most $l(k)$ other clauses the formula is satisfiable.

With this notation the Lovász Local Lemma implies that

$$(1.1) \quad l(k) \geq \left\lfloor \frac{2^k}{e} \right\rfloor - 1.$$

The order of magnitude of this bound is trivially optimal: $l(k) < 2^k - 1$ follows from the unsatisfiable k -CNF consisting of all possible k -clauses on only k variables.

In [9] a hardness jump is proved for the function l : deciding the satisfiability of k -CNF formulas with maximum neighborhood size at most $\max\{l(k) + 2, k + 3\}$ is NP-complete.

As observed by Kratochvíl, Savický and Tuza [15] the bound (1.1) immediately implies

$$(1.2) \quad f(k) \geq \left\lfloor \frac{l(k)}{k} \right\rfloor + 1 \geq \left\lfloor \frac{2^k}{ek} \right\rfloor.$$

From the other side Savický and Sgall [21] showed that $f(k) = O\left(k^{0.74} \cdot \frac{2^k}{k}\right)$. This was improved by Hoory and Szeider [12] who came within a logarithmic factor: $f(k) = O\left(\log k \cdot \frac{2^k}{k}\right)$. Recently Gebauer [10] showed that the order of magnitude of the lower bound is correct and $f(k) = \Theta(2^k/k)$. This construction also improved the trivial upper bound on $l(k)$ showing $l(k) \leq \frac{63}{128}2^k$ for infinitely many k .

In our main theorem we determine the asymptotics of both $f(k)$ and $l(k)$. We show that the lower bound (1.1) on $l(k)$ is asymptotically tight but its consequence (1.2) is not tight, it can be strengthened by a factor of 2.

Theorem 1.1.

$$\left\lfloor \frac{2^{k+1}}{ek} \right\rfloor - 1 \leq f(k) = \left(e^{-1} + O(k^{-1/2})\right) \frac{2^{k+1}}{k}.$$

Theorem 1.2.

$$\left\lfloor \frac{2^k}{e} \right\rfloor - 1 \leq l(k) = \left(e^{-1} + O(k^{-1/2})\right) 2^k.$$

The lower bound was achieved via the lopsided version of the Lovász Local Lemma by Berman, Karpinski and Scott [2], although they did not calculate the asymptotic behavior of their bound.

Since the (lopsided) Lovász Local Lemma was fully algorithmized by Moser and Tardos [17] we now have that not only every (k, s) -CNF formula for $s = \lfloor 2^{k+1}/(ek) \rfloor - 1$ has a satisfying assignment but there is also an algorithm that *finds* such an assignment in probabilistic polynomial time. Moreover, for just a little bit larger value of the parameter s one cannot find a satisfying assignment efficiently simply because already the decision problem is NP-hard.

1.1. The class MU(1). The function $f(k)$ is not known to be computable. In order to still be able to upper bound its value, one tries to restrict to a smaller/simpler class of formulas. When looking for unsatisfiable (k, s) -CNF formulas it is naturally enough to consider *minimal unsatisfiable formulas*, i.e., unsatisfiable CNF formulas that become satisfiable if we delete any one of their clauses. The set of minimal unsatisfiable CNF formulas is denoted by MU. As observed by Tarsi (c.f. [1]), all formulas in MU have more clauses than variables, but some have only one more. The

class of these MU formulas, having one more clauses than variables is denoted by MU(1). This class has been widely studied (see, e.g., [1], [4], [13], [16], [23]). Hoory and Szeider [11] considered the function $f_1(k)$, denoting the largest integer such that no $(k, f_1(k))$ -CNF formula is in MU(1). They showed that $f_1(k)$ is computable and is an upper bound for $f(k)$. Their computer search determined the value of $f_1(k)$ for small k : $f_1(5) = 7$, $f_1(6) = 11$, $f_1(7) = 17$, $f_1(8) = 29$, $f_1(9) = 51$. Via the trivial inequality $f(k) \leq f_1(k)$, these are the best known upper bounds on $f(k)$ in this range. In contrast, even the value of $f(5)$ is not known.

It is also not known whether $f(k) = f_1(k)$ for every k . Our upper bound construction from Theorem 1.1 does reside in the class MU(1) and hence shows that $f(k)$ and $f_1(k)$ are equal asymptotically: $f(k) = (1 + o(1))f_1(k)$.

Scheder [22] showed that for almost disjoint k -CNF formulas (i.e. CNF-formulas where any two clauses have at most one variable in common) the two functions are not the same. That is, if $\tilde{f}(k)$ denotes the maximum s such that every almost disjoint (k, s) -CNF formula is satisfiable, for k large enough every unsatisfiable almost disjoint $(k, \tilde{f}(k) + 1)$ -CNF formula is outside of MU(1).

2. CONSTRUCTING BINARY TREES AND MU(1) FORMULAS

The structure of MU(1) formulas is well understood and it is closely related to binary trees. In particular, given any finite rooted binary tree T with the property that each non-leaf vertex has exactly two children we associate with it certain CNF formulas. We start with assigning distinct literals to the vertices, assigning the negated and non-negated form of the same variable to the two children of any non-leaf vertex. We do not assign any literal to the root. For each leaf of T consider a clause that is the disjunction of some literals along the path from the root to that leaf and consider the CNF formula F that is the conjunction of one clause for each leaf. Clearly, F is unsatisfiable and it has one more clauses than the number of variables associated with T . As proved in [4] F is a MU(1) formula if and only if all literals associated to vertices of T do appear in F , furthermore every formula in MU(1) can be obtained from a suitable binary tree this way.

We bound $f(k)$ through bounding $f_1(k)$, that is by finding (k, d) -CNF formulas in MU(1). By the above correspondence between MU(1) formulas and trees this involves constructing a rooted binary tree and selecting a k -subset of the vertices on every root-leaf path. We want to concentrate on the tree building so we further restrict the class of k -CNF formulas in MU(1) that we are considering to formulas obtained from binary trees by selecting on every root-leaf path the k vertices farthest from the root. Accordingly we define:

A (k, d) -tree is a finite rooted binary tree with all non-leaf vertices having exactly two children that satisfy that (i) there is no leaf in distance less than k from the root and (ii) for every vertex v there is at most d leaves among the descendants of v in distance at most k of v .

Let $f_2(k)$ be the largest integer d such that no (k, d) -tree exist. We clearly have

Lemma 2.1. $f_1(k) \leq f_2(k)$ for every k .

Proof: Let $d = f_2(k) + 1$, T a (k, d) -tree and consider the associated unsatisfiable k -CNF formula F as explained above. Note that condition (i) in the definition ensures

that each root-leaf path is long enough and so F exists, while condition (ii) makes F a (k, d) -CNF. The existence of F proves $f(k) < d$, so we have $f(k) \leq f_2(k)$. To prove $f_1(k) \leq f_2(k)$ we need a (k, d) -CNF in $\text{MU}(1)$. Unfortunately, F does not necessarily lie in $\text{MU}(1)$. This is because if some literal associated to a vertex v of T does not appear in F , then F is not even in MU . This happens if and only if there is no leaf in distance less than k among the descendants of the non-leaf vertex v . In this case the subtree of T rooted at v is also a (k, d) -tree, so if we pick T to be a minimal (k, d) -tree, then the corresponding (k, d) -CNF formula is in $\text{MU}(1)$. \square

Our goal is to bound $f_2(k)$ (and hence $f_1(k)$ and $f(k)$) by constructing (k, d) -trees for suitable values of k and d .

A (k, d) -vector is a $(k + 1)$ -tuple $x = (x_0, \dots, x_k)$ of non-negative integers with $|x| \leq d$. Here $|x|$ is defined as $|x| = \sum_{j=0}^k x_j$, while the *weight* of x is $\sum_{j=0}^k x_j / 2^j$. We call a finite rooted binary tree a (k, d, x) -tree if all non-leaf vertices have exactly two children and (i') there is at most x_j leaves in distance exactly j of the root for $j = 0, \dots, k$ and (ii) for every vertex v there is at most d leaves among the descendants of v in distance at most k of v . We call a (k, d) -vector x (k, d) -constructible, or *constructible* if k and d are clear from the context, if a (k, d, x) -tree exists.

We will need the following lemma, which is a dual version of Kraft's inequality [14].

Lemma 2.2. *For a (k, d) -vector x a (k, d, x) -tree of depth at most k exists if and only if the weight of x is at least 1. In particular, the (k, d) -vectors of weight at least 1 are constructible.*

Note that to obtain this form of Kraft's inequality from the standard version (a detailed explanation can be found e.g., in [20], Theorem 2.1.2.) one has to use the simple observation that if the weight of x is strictly above 1, then there is another (k, d) -vector x' coordinate-wise dominated by x , whose weight is exactly 1. A very similar form of the inequality was also used in [10].

Let us fix the positive integers k, d and l . We will not show the dependence on these parameters in the next definitions to simplify the notation, although d', E, C_r and C_r^* depend on them. We let $d' = d(1 - 1/(2^l - 1))$. For a (k, d) -vector $x = (x_0, \dots, x_k)$ we define $E(x) = (\lfloor x_1/2 \rfloor, \lfloor x_2/2 \rfloor, \dots, \lfloor x_k/2 \rfloor, \lfloor d'/2 \rfloor)$. We denote by $E^m(x)$ the vector obtained from x by m applications of the operation E . For $l \leq r \leq k$ we define $C_r(x)$ to be the $(k + 1)$ -tuple starting with $r + 1 - l$ zero entries followed by $\lfloor x_j/(2^l - 1) \rfloor$ for $j = r + 1, \dots, k$, followed by $\lfloor d'/2^{l-j} \rfloor$ for $j = 0, 1, \dots, l - 1$ and let $C_r^*(x)$ be the $(k + 1)$ -tuple starting with x_j for $j = l, l + 1, \dots, r$ followed by $k - r + l$ zeros.

Note that for the following lemma to hold we could use d instead of d' in the definition of E and also in most places in the definition of C . The one place where we cannot do this is the entry $\lfloor d'/2^l \rfloor$ of $C_r(x)$ right after $\lfloor x_k/(2^l - 1) \rfloor$. If we used a higher value there, then one of the children of the root of the tree constructed in the proof below would have more than d leaves among its descendants in distance at most k . We use d' everywhere to be consistent and provide for the monotonicity as used in the proof of Theorem 2.1.

Lemma 2.3. *Let k, d and l be positive integers and x a (k, d) -vector.*

(a) *$E(x)$ is a (k, d) -vector. If $E(x)$ is constructible, then so is x .*

(b) For $l \leq r \leq k$ both $C_r(x)$ and $C_r^*(x)$ are (k, d) -vectors. If both of these vectors are constructible and $|C_r^*(x)| < d/2^l$, then x is also constructible.

Proof: (a) We have $|E(x)| \leq |x|/2 + d'/2 < d$, so $E(x)$ is a (k, d) -vector. If there exists a $(k, d, E(x))$ -tree, take two disjoint copies of such a tree and connect them with a new root vertex, whose children are the roots of these trees. The new binary tree so obtained is a (k, d, x) -tree.

(b) The sum of the first $k+1-l$ entries of $C_r(x)$ is at most $|x|/(2^l-1) \leq d/(2^l-1)$, the remaining fixed terms sum to less than $d' = d(1 - 1/(2^l-1))$, so $|C_r(x)| \leq d$. We trivially have $|C_r^*(x)| \leq |x| \leq d$, so both $C_r(x)$ and $C_r^*(x)$ are (k, d) -vectors.

Let T be a $(k, d, C_r(x))$ -tree and T^* a $(k, d, C_r^*(x))$ -tree. Consider a full binary tree of depth l and attach T^* to one of the 2^l leaves of this tree and attach a separate copy of T to all remaining $2^l - 1$ leaves. This way we obtain a finite binary tree T' with all non-leaf vertices having exactly two children. To check condition (i') of the definition notice that no leaf of T' is in distance less than l from the root, leaves in distance $l \leq j \leq r$ are all in T^* and leaves in distance $r < j \leq k$ are all in the $2^l - 1$ copies of T . Condition (ii) we have to check only for vertices in distance $1 \leq j \leq l$ from the root. There are two types of vertices in distance j . One of them has 2^{l-j} copies of T below it, the other has one less and also T^* . It is a straight forward calculation to bound the number of leaves below and in distance at most k in either case. Instead of giving all the details of the computation we mention that the vertex of the first type in distance $j = 1$ of the root makes us use the specific value of $d' < d$ we use and the vertices of the second type make it necessary to assume a bound on $|C_r^*(x)|$. \square

Armed with the last two lemmas we are ready to prove the main result of this paper.

Theorem 2.1.

$$f_2(k) \leq \frac{2^{k+1}}{ek} + O\left(\frac{2^{k+1}}{k^{3/2}}\right)$$

We show that (k, d) -trees exist for large enough k and with $d = \lfloor 2^{k+1}/(ek) + 100 \cdot 2^{k+1}/k^{3/2} \rfloor$.

We set $l = \lfloor \log k/2 \rfloor$ and $s = 2l$. Here \log denotes the binary logarithm, so $2^l \approx \sqrt{k}$. We define the (k, d) -vectors $x^{(t)} = (x_0^{(t)}, \dots, x_k^{(t)})$ recursively. We start with $x^{(0)} = E^{k-s}(z)$, where z denotes the all zero (k, d) -vector. For $t \geq 0$ we define $x^{(t+1)} = E^{r_t-s-l}(C_{r_t}(x^{(t)}))$, where r_t is the smallest index in the range $s+l \leq r_t \leq k$ with $\sum_{j=0}^{r_t} x_j^{(t)}/2^j \geq 2^{-l}$. At this point we may consider the sequence of the (k, d) -vectors $x^{(t)}$ and whenever the weight of one of them falls below 2^{-l} and thus the definition of r_t does not make sense. But we will establish below that this never happens and the sequence is infinite.

Notice first, that we have $x_j^{(t)} = 0$ for all t and $0 \leq j \leq s$, while the entries $x_j^{(t)}$ for $s < j \leq k$ are all obtained from d' by repeated application of dividing by an integer (namely by $2^l - 1$ or by a power of 2) and taking lower integer part. Using the simple observation that $\lfloor \lfloor a \rfloor / j \rfloor = \lfloor a/j \rfloor$ if a is real and j is a positive integer we can ignore all roundings but the last. This way we can write each of these entries

in the form $\left\lfloor \frac{d'}{2^i(2^l-1)^j} \right\rfloor = \left\lfloor \frac{d'}{2^{i+lj}} \left(1 + \frac{1}{2^l-1}\right)^j \right\rfloor$ for some non-negative integers i and j . Using the values $\alpha = 1 + 1/(2^l - 1)$ and $q_t = r_t - s$ (this is the amount of “left shift” between x^t and x^{t+1}) we can give the exponents explicitly.

$$(2.1) \quad x_j^{(t)} = \left\lfloor \frac{d'}{2^{k+1-j}} \alpha^{c(t,j)} \right\rfloor$$

for all $s < j \leq k$ and all t , where $c(t, j)$ is the largest integer $0 \leq c \leq t$ satisfying $\sum_{i=t-c}^{t-1} q_i \leq k - j$. We define $c(t, j) = 0$ if $q_{t-1} > k - j$.

The formal inductive proof of this formula is a straight forward calculation. What really happens here (ignoring the rounding) is that each entry enters at the right end of the vector as $d'/2$ and is divided by 2 every time it moves one place to the left (application of E) but when it moves l places to the left through an application of C_{r_t} it is divided by $2^l - 1$ instead of 2^l so it gains a factor of α . The exponent $c(t, j)$ counts how many such factors are accumulated. If the “ancestor” of the entry $x_j^{(t)}$ was first introduced in $x^{(t')}$, then $c(t, j) = t - t'$.

We claim next that $c(t, j)$ and $x_j^{(t)}$ increases monotonously in t for each fixed $s < j \leq k$, while q_t decreases monotonously with t . We prove these statements by induction on t . We have $c(0, j) = 0$ for all j , so $c(1, j) \geq c(0, j)$. If $c(t+1, j) \geq c(t, j)$ for all j , then all entries of $x^{(t+1)}$ dominate the corresponding entries of $x^{(t)}$ by Equation (2.1). If $x_j^{(t+1)} \geq x_j^{(t)}$ for all j , then we have $r_{t+1} \leq r_t$ by the definition of these numbers, so we also have $q_{t+1} \leq q_t$. Finally, if q_i is decreasing for $i \leq t+1$, then by the definition of $c(i, j)$ we have $c(i+1, j) \geq c(i, j)$ for $i \leq t+1$.

The monotonicity just established also implies that the weight of $x^{(t)}$ is also increasing, so if the weight of $x^{(0)}$ is at least 2^{-l} , then so is the weight of all the other $x^{(t)}$, and thus the sequence is infinite. The weight of $x^{(0)}$ is

$$\begin{aligned} &= \sum_{j=s+1}^k \frac{\left\lfloor \frac{d'}{2^{k+1-j}} \right\rfloor}{2^j} \\ &> \sum_{j=s+1}^k \frac{\frac{d'}{2^{k+1-j}} - 1}{2^j} \\ &> (k-s) \frac{d'}{2^{k+1}} - 2^{-s} \\ &> (k-s) \frac{1 - \frac{1}{2^l-1}}{ek} - 2^{-s}, \end{aligned}$$

where the last inequality follows from $d > 2^{k+1}/(ek)$ and the last term tends to e^{-1} as k tends to infinity, so it is larger than 2^{-l} for large enough k .

We have just established that the sequence $x^{(t)}$ of (k, d) -vectors is infinite and coordinate-wise increasing. Since $|x^{(t)}| \leq d$ and it must strictly increase before the sequence stabilizes, the sequence must stabilize in at most d steps. So $x^{(t)} = x = (x_0, \dots, x_k)$ for $t \geq d$. This implies that q_t also stabilizes with $q_t = q$ for $t \geq d$.

Equation (2.1) as applied to $t > d + k$ simplifies to

$$(2.2) \quad x_j = \left\lfloor \frac{d'}{2^{k+1-j}} \alpha^{\lfloor (k-j)/q \rfloor} \right\rfloor.$$

Recall that $q = r_{t_0} - s$, and r_{t_0} is defined as the smallest index in the range $s + l \leq r \leq k$ with $\sum_{j=0}^r x_j/2^j \geq 2^{-l}$. Thus we have $q \geq l$. We claim that equality holds. Assume for contradiction that $q > l$. Then by the minimality of r_{t_0} we must have

$$\begin{aligned} 2^{-l} &> \sum_{j=0}^{s+q-1} \frac{x_j}{2^j} \\ &= \sum_{j=s+1}^{s+q-1} \frac{\left\lfloor \frac{d'}{2^{k+1-j}} \alpha^{\lfloor (k-j)/q \rfloor} \right\rfloor}{2^j} \\ &> \sum_{j=s+1}^{s+q-1} \frac{\frac{d'}{2^{k+1-j}} \alpha^{(k-j)/q-1} - 1}{2^j} \\ &> (q-1) \frac{d'}{2^{k+1}} \alpha^{k/q-4} - 2^{-2l}. \end{aligned}$$

In the last inequality we used $s = 2l$ and that $\frac{j}{q} \leq \frac{s+q}{q} = 1 + \frac{s}{q} \leq 1 + \frac{2l}{l} = 3$. This inequality simplifies to

$$(2.3) \quad 2^{-l}(1 + 2^{-l})\alpha^4 \frac{2^{k+1}}{d'} > (q-1)\alpha^{k/q}.$$

Simple calculus gives that the right hand side takes its minimum for $q \geq 2$ between $c-1$ and $c-2$ for $c = k \ln \alpha$ and this minimum is more than $(c-3)e$. Using $\alpha = 1 + 1/(2^l - 1) > e^{2^{-l}}$ we have $c \geq k/2^l$. So Inequality (2.3) yields

$$2^{-l}(1 + 2^{-l})\alpha^4 \frac{2^{k+1}}{d'} > \frac{ke}{2^l} - 3e.$$

Substituting our choice for d, d', α and l (as functions of k) shows that this last formula does not hold for large k . The contradiction proves $q = l$ as claimed.

We finish the proof of the theorem by establishing that the (k, d) -vectors $x^{(t)}$ are constructible. We prove this statement by downward induction for t . We start with $t = d$, where $x^{(d)} = x$ and by Equation (2.2) its weight is readily computable. Here we omit the details of this straight forward calculation but state that the weight is above 1. So by Lemma 2.2 x is constructible.

Now assume that $x^{(t+1)}$ is constructible. Recall that $x^{(t+1)} = E^{r_t-s-l}(C_{r_t}(x^{(t)}))$, so by (the repeated use of) Lemma 2.3 part (a) $C_{r_t}(x^{(t)})$ is constructible. By part (b) of the same lemma $x^{(t)}$ is also constructible (and thus the inductive step is complete) if we can (i) show that $C_{r_t}^*(x^{(t)})$ is constructible and (ii) establish that $|C_{r_t}^*(x^{(t)})| \leq d/2^l$. For (i) we use the definition of r_t : $\sum_{j=0}^{r_t} x_j^{(t)}/2^j \geq 2^{-l}$. But the weight of $C_{r_t}^*(x^{(t)})$ is $\sum_{j=l}^{r_t} x_j^{(t)}/2^{j-l}$, so the contribution of each term with $j \geq l$ is multiplied by 2^l , while the missing terms $j < l$ contributed zero anyway as $l < s$. This shows that the weight of $C_{r_t}^*$ is at least 1 and therefore Lemma 2.2

proves (i). For (ii) we use monotonicity to see $r_t \leq r_0$ and Equation (2.1) to see that $r_0 \leq 5k/2^l$ for large enough k . Then we use monotonicity again to see $|C_{r_t}^*(x^{(t)})| = \sum_{j=s+1}^{r_t} x_j^{(t)} \leq \sum_{j=s+1}^{r_t} x_j$. Finally we use Equation (2.2) to see that this number is well below $d/2^l$ for large enough k . This finishes the proof of (ii) and hence the inductive proof that $x^{(t)}$ is constructible for every t .

As $x^{(0)} = E^{k-s}(z)$ is constructible Lemma 2.3 (a) implies that z is also constructible, so there exists (k, d, z) -tree T . As z is the all zero vector, T must also be a (k, d) -tree. \square

3. PROOF OF THEOREMS 1.1 AND 1.2

Proof of Theorem 1.2: The lower bound part of the theorem is an immediate and standard application of the the Lovász Local Lemma. For the upper bound we need unsatisfiable k -CNF formulas with each clause intersecting just a few other clauses. Note that the formula constructed in the previous proof is not necessarily good enough. We need not only that each of the variables appears at most d times, but the variables must also be balanced: no literal should appear significantly more than $d/2$ times. Fortunately the existence of such formulas follows easily from Theorem 2.1 and Lemma 3.1 below. We remark here that we do not have to rely on the fact that the proof of Theorem 2.1 established not only the existence of (k, d) -trees for $d = (1 + O(k^{-1/2}))2^{k+1}/(ek)$ but also the existence (k, d, z) -trees, where z is the all zero vector. The latter readily follows from the former, simply take two copies of a (k, d) -tree and join them by a new root vertex whose children are the roots of the two copies. The tree so obtained is a (k, d, z) -tree. \square

Lemma 3.1. *Let T be a (k, d, z) -tree, where z is the all zero (k, d) -vector. Then T is also a $(k+1, 2d)$ -tree and in the unsatisfiable $(k+1)$ -CNF formula F corresponding to T every literal appears at most d times. Furthermore every clause of F intersects at most $d(k+1)$ other clauses.*

Proof: The first statement is immediate from the correspondence between trees and formulas. For the last statement note that a formula so obtained from a tree has the property that no two of its clauses are positively correlated: if they intersect they have a variable that appears positively in one and in negated form in the other. Therefore each other clause that a given clause C intersects has a literal that is opposite to a literal appearing in C . \square

Berman, Karpinski and Scott [2] noticed that the lower bound (1.2) on $f(k)$ that is implied by the Lovász Local Lemma can be improved using the lopsided version of the lemma. They do not compute the asymptotic implications of their bound because they believe that “it should be possible to obtain better bounds by employing techniques from hypergraph coloring (see, for instance, Radhakrishnan and Srinivasan [19]).”

As it turns out their lower bound improves (1.2) by a factor of 2 and is asymptotically tight. It is proved with the lopsided version of the local lemma that allows for “ignoring” that a pair of clauses in a CNF formula intersects if no variable appears negated in one and in non-negated form in the other. This reduces the number of intersections to consider and if each variable appears at most $d/2$ times negated and at most $d/2$ times non-negated in a (k, d) -CNF, then each clause intersects at

most $kd/2$ other clauses instead of the original $k(d-1)$ bound. If, however, some of the variables are unbalanced and say appear almost d times negated but only a handful of times non-negated, then there might be a few clauses that still intersect almost kd other clauses. This situation is handled by the non-uniform version of the local lemma and by making these variables biased. Surprisingly, however, one has to make a variable biased in a very counter-intuitive direction: the more the variable appears negated the less likely we set it false. (This strange direction of the bias is not apparent from [2], probably due to a typo.)

Theorem 3.1 ([2]). *We have $f(k) \geq d$ if there exists $x \in (0, 1)$ with*

$$x^{1/k}((1-x)^{\lceil d/2 \rceil} + (1-x)^{\lfloor d/2 \rfloor}) \geq 1.$$

Proof of Theorem 1.1: Let k be an arbitrary positive integer, $d = \left\lfloor \frac{2^{k+1}}{ek} \right\rfloor - 1$ and set $x = \frac{e}{2^k}$. We have

$$x^{1/k}((1-x)^{\lceil d/2 \rceil} + (1-x)^{\lfloor d/2 \rfloor}) \geq 2x^{1/k}(1-x)^{d/2} \geq e^{1/k}(1 - \frac{e}{2^k})^{\frac{2^k}{ek} - \frac{1}{2}} > 1.$$

Thus Theorem 3.1 establishes the lower bound part of the theorem. The upper bound follows from Theorem 2.1, Lemma 2.1 and the fact that $f(k) \leq f_1(k)$. \square

4. ON THE SIZE OF UNSATISFIABLE FORMULAS

By the *size* of a tree we mean the number of its leaves and by the *size* of a CNF formula we mean the number of its clauses. With this notation the size of a (k, d) -tree and the size of the corresponding (k, d) -CNF in MU(1) are the same.

When proving Theorem 2.1 we constructed (k, d) -trees for $d \approx \frac{2^{k+1}}{ek}$. Their size and therefore the size of the corresponding (k, d) -CNF in MU(1) is at most 2^h , where if h is the *depth* of the tree: the largest root-leaf distance. In fact, the size of the trees we constructed are very close to this upper bound. Therefore it makes sense to take a closer look at the depth.

Let us associate with a vertex v of a (k, d) -tree the (k, d) -vector (x_0, \dots, x_k) , where x_j is the number of leaves in distance j from v among the descendants of v . A minimal (k, d) -tree has no two vertices along the same branch with identical vectors, so the depth of a minimal (k, d) -tree is limited by the number of (k, d) -vectors, less than $(d+1)^{k+1}$. For $d = f(k) + 1$ this is $2^{\Theta(k^2)}$. The same bound for minimal (k, d) -CNF formulas in MU(1) is implicit in [11]. There is numerical evidence that the size of the minimal $(k, f(k) + 1)$ -tree and the minimal (k, d) -CNF in MU(1) might indeed be doubly exponential in k (consider the size of the minimal $(7, 18)$ -tree and the minimal $(7, 18)$ -CNF in MU(1) mentioned below).

A closer analysis of the proof of Theorem 2.1 shows that the depth of the (k, d) -tree constructed in it is at most kd . While this is better than the general upper bound above it still allows for trees with sizes that are doubly exponential in k .

This depth can, however, be substantially decreased if we allow the error term in d to slightly grow. If we allow $d = (1 + \epsilon)\frac{2^{k+1}}{ek}$ for a fixed $\epsilon > 0$, then a more careful analysis shows that the depth of the tree created becomes $O(k)$. This bounds the size of the tree and the corresponding formula by a polynomial of d . Here we just sketch the argument for this statement and do not give a formal proof. The main point is

to note that we can stop the recursion defining $x^{(t)}$ in the proof of Theorem 2.1 as soon as the weight of one of these vectors reaches 1 and to give an accurate estimate of how fast this will happen. For this we define a continuous two-variable function $F(T, X)$ on $T \geq 0$ and $X \in [0, 1]$ and use it to approximate the vectors $x^{(t)}$. Let $Q_t = \sum_{i=0}^{t-1} q_i$ for the integers q_t defined alongside $x^{(t)} = (x_0^{(t)}, \dots, x_k^{(t)})$ in the same proof. We show that

$$x_j^{(t)} \approx \frac{d}{2^{k+1-j}} F\left(\frac{Q_t}{k}, \frac{j}{k}\right).$$

The function F is given by the boundary conditions $F(0, X) = 1$ for $X \in [0, 1]$, $F(T, 1) = 1$ for $T \geq 0$ and by the differential equation

$$D_{(1, -1)} F(T, X) = \beta F(T, X) F(T, 0),$$

where $D_{(1, -1)}$ stands for the derivative in the direction $(1, -1)$, that is $\frac{\partial}{\partial T} - \frac{\partial}{\partial X}$ and $\beta = dk/2^{k+1}$ is a constant. Analyzing the differential equation one finds that if $\beta \leq e^{-1}$, then $F(T, X)$ remains bounded by e^{1-X} , but for $\beta > e^{-1}$ it reaches infinity fast. Naturally F fails to approximate the discrete process at this point of singularity, but somewhat before that point we find a corresponding $x^{(t)}$ vector with large weight. The corresponding value Q_t is $O(k)$ and thus the (k, d) -tree obtained will have depth $O(k)$ too.

Let us define $f_1(k, d)$ for $d > f_1(k)$ to be the size of the smallest (k, d) -CNF in MU(1) and let $f_2(k, d)$ stand for the size of the smallest (k, d) -tree, assuming $d > f_2(k)$. While $f_1(k, f_1(k) + 1)$ and similarly $f_2(k, f_2(k) + 1)$ are probably doubly exponential in k , for slightly larger values of d $f_1(k, d)$ and $f_2(k, d)$ are polynomial in d (and thus simply exponential in k).

Finally we mention the question whether $f_1(k) = f_2(k)$ for all k . In other words, we ask whether we lose by selecting the k vertices farthest from the root when making a MU(1) k -CNF formula from a binary tree. It should be mentioned that $f(k) = f_1(k)$ is also open, but $f_1(k) = f_2(k)$ seems to be a simpler question as both functions are computable. Computing their values up to $k = 7$ we found these values agreed. To gain more insight we computed the corresponding size functions too and found that $f_1(k, d) = f_2(k, d)$ for $k \leq 6$ and all $d > f_1(k)$. But for $k = 7$, where $f_1(7) = f_2(7) = 17$ we found the following surprising result: $f_1(7, d) = f_2(7, d)$ for $19 \leq d \leq 25$, but $f_1(7, 18) = 10, 197, 246, 480, 846$, while $f_2(7, 18) = 10, 262, 519, 933, 858$. In other words, there exist $(7, 18)$ -trees, but the smallest $(7, 18)$ -CNF formulas cannot be obtained from them. Does this indicate that all other equalities are coincidences and f_1 and f_2 will eventually diverge?

A related algorithmic question is whether the somewhat simpler structure of (k, d) -trees can be used to find an algorithm computing $f_2(k)$ substantially faster than the algorithm of Hoory and Szeider [11] for computing $f_1(k)$. Such an algorithm would give useful estimates for $f_1(k)$ and also $f(k)$. At present we use a similar (and similarly slow) algorithm for either function.

REFERENCES

- [1] R. Aharoni and N. Linial, Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas, *J. Combin. Theory Ser. A* **43**, (1986), 196–204.

- [2] P. Berman, M. Karpinski, and A. D. Scott, Approximation hardness and satisfiability of bounded occurrence instances of SAT. *Electronic Colloquium on Computational Complexity (ECCC)*, **10** (022), 2003.
- [3] S.A. Cook, The complexity of theorem-proving procedures, *Proc. 3rd Ann. ACM Symp. on Theory of Computing (STOC)* (1971), 151–158.
- [4] G. Davydov, I. Davydova, and H. Kleine Büning, An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF, *Artif. Intell.* **23**, (1998), 229–245.
- [5] O. Dubois, On the r, s -SAT satisfiability problem and a conjecture of Tovey, *Discrete Appl. Math.* **26** (1990), 51–60.
- [6] P. Erdős and J. Spencer, Lopsided Lovász local lemma and Latin transversals *Discrete Appl. Math.* **30**, (1991), 151–154.
- [7] U. Feige, A threshold of $\ln n$ for approximating set cover, *J. ACM* **45**(4), (1998), 634–652.
- [8] C.M. Fortuin, P.N. Kasteleyn, J. Ginibre, Correlation inequalities for some partially ordered sets, *Comm. Math. Phys.* **22** (1971), 89–103
- [9] H. Gebauer, R. A. Moser, D. Scheder and E. Welzl, The Lovász Local Lemma and Satisfiability *Efficient Algorithms*, (2009), 30–54.
- [10] H. Gebauer, Disproof of the Neighborhood Conjecture with Implications to SAT, *Proc. 17th Annual European Symposium on Algorithms (ESA)* (2009), LNCS 5757, 764–775.
- [11] S. Hoory and S. Szeider. Computing unsatisfiable k -SAT instances with few occurrences per variable, *Theoretical Computer Science* **337**(1–3) (2005), 347–359,
- [12] S. Hoory and S. Szeider, A note on unsatisfiable k -CNF formulas with few occurrences per variable, *SIAM J. Discrete Math* **20** (2), (2006), 523–528.
- [13] H. Kleine Büning and X. Zhao, On the structure of some classes of minimal unsatisfiable formulas, *Discr. Appl. Math.* **130**(2), (2003), 185–207
- [14] L. G. Kraft, A device for quantizing, grouping, coding amplitude modulated pulses, *M.S. thesis, Electrical Engineering Department, MIT* (1949).
- [15] J. Kratochvíl, P. Savický and Z. Tuza, One more occurrence of variables makes satisfiability jump from trivial to NP-complete, *SIAM J. Comput.* **22** (1), (1993), 203–210.
- [16] O. Kullmann, An application of matroid theory to the SAT problem, *Fifteenth Annual IEEE Conference on Computational Complexity* (2000), 116–124
- [17] R.A. Moser, G. Tardos, A constructive proof of the general lovsz local lemma, *J. ACM* **57**(2), (2010)
- [18] C.H. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *J. Comput. System Sci.* **43** (3), (1991), 425–440
- [19] J. Radhakrishnan, A. Srinivasan, Improved bounds and algorithms for hypergraph 2-coloring, *Random Structures Algorithms* **16** (3), (2000), 4–32
- [20] S. Roman, Coding and Information Theory, *Springer, New York* (1992).
- [21] P. Savický and J. Sgall, DNF tautologies with a limited number of occurrences of every variable, *Theoret. Comput. Sci.* **238** (1–2), (2000), 495–498.
- [22] D. Scheder, Existence, Size, and Resolution Complexity of Almost Disjoint CNF Formulas, *submitted*
- [23] S. Szeider, Homomorphisms of conjunctive normal forms, *Discr. Appl. Math.* **130**(2), (2003), 351–365
- [24] C.A. Tovey, A simplified NP-complete satisfiability problem, *Discr. Appl. Math.* **8** (1), (1984), 85–89.